Session Type Systems based on Linear Logic

Heuvel, Bas van den; Pérez, Jorge A.

Published in:
Electronic Notes in Theoretical Computer Science

DOI:
10.4204/EPTCS.314.1

Publication date:
2020

Session Type Systems based on Linear Logic: Classical versus Intuitionistic*

Bas van den Heuvel Jorge A. Pérez
University of Groningen, The Netherlands

Session type systems have been given logical foundations via Curry-Howard correspondences based on both intuitionistic and classical linear logic. The type systems derived from the two logics enforce communication correctness on the same class of π-calculus processes, but they are significantly different. Caires, Pfenning, and Toninho informally observed that, unlike the classical type system, the intuitionistic type system enforces locality for shared channels, i.e. received channels cannot be used for replicated input. In this paper, we revisit this observation from a formal standpoint. We develop United Linear Logic (ULL), a logic encompassing both classical and intuitionistic linear logic. Then, following the Curry-Howard correspondences for session types, we define πULL, a session type system for the π-calculus based on ULL. Using πULL we can formally assess the difference between the intuitionistic and classical type systems, and justify the role of locality and symmetry therein.

1 Introduction

Session types are a popular approach to typing message-passing concurrency [10, 11, 17]. They describe communication over channels as sequences of communication actions. This way, e.g., the session type !int.?bool.end types a channel as follows: send an integer, receive a boolean, and close the channel. Due to its simplicity and expressiveness, the π-calculus [15, 16]—the paradigmatic model of concurrency and interaction—is a widely used setting for studying session types.

In a line of work developed by Caires, Pfenning, Wadler, and several others, the theory of session types has been given strong logical foundations. Caires and Pfenning discovered a Curry-Howard correspondence between a form of session types for the π-calculus and Girard's *linear logic* [8]: session types correspond to linear logic propositions, type inference rules to sequent calculus, and communication to cut reduction [2]. The resulting session type system ensures important correctness properties for communicating processes: protocol fidelity, communication safety, deadlock freedom, and strong normalization.

As in standard logic, there are two "schools" of linear logic: *classical* [8] and *intuitionistic* [1]. The differences between classical and intuitionistic linear logic are known—see, e.g., [5, 12]. This dichotomy also appears in the logical foundations of session types: while Caires and Pfenning's correspondence relies on intuitionistic linear logic [1], Wadler developed a correspondence based on classical linear logic [18]. Superficial differences between the resulting type systems include the number of typing rules (the intuitionistic system has roughly twice as many rules as the classical system) and the shape/meaning of typing judgments (in the intuitionistic system, judgments have a rely-guarantee reading not present in the classical system). In turn, these differences follow from the way in which each system internalizes duality: the classical system provides a more explicit account of duality than the intuitionistic system.

*Work partially supported by the Netherlands Organization for Scientific Research (NWO) under the VIDI Project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

S. Balzer, L. Padovani (Eds.): Programming Language Approaches to Concurrency- & Communication-cEntric Software (PLACES 2020)
EPTCS 314, 2020, pp. 1–11, doi:10.4204/EPTCS.314.1

We are interested in going beyond these superficial differences, so as to establish a formal comparison between the two type systems. This seems to us an indispensable step in consolidating the logical foundations of message-passing concurrency. To our knowledge, the only available comparison is informal: Caires, Pfenning, and Toninho [4] observed that a more fundamental difference concerns the *locality principle* for shared channels. The principle states that received channels cannot be used for further reception, i.e., only the output capability of channels can be sent [13]. In session-based concurrency, shared channels define services; clients connect to services by sending a linear channel. Locality of shared channels therefore means that received channels cannot be used to provide a service. Well-known from a foundational perspective, locality has been promoted as a sensible principle for distributed implementations of (object-oriented) languages based on the π-calculus [14]. The observation in [4] is that Caires and Pfenning's intuitionistic interpretation of session types enforces locality of shared channels [4], whereas Wadler's classical interpretation does not: processes that break locality are well-typed in [18].

The existence of a class of processes that is typable in one system but not in the other immediately frames the desired formal comparison as an expressiveness question: the type system in [18] can be considered to be *more expressive* than the one in [2]. To formally examine this question, the first step is defining a basic framework of reference in which both type systems can be objectively compared. To this end, we build upon Girard's Logic of Unity [9], which subsumes classical, intuitionistic and linear logic in one system. In the same spirit, we develop *United Linear Logic* (ULL): a logic that subsumes classical and intuitionistic linear logic. Following the Curry-Howard correspondence by Caires and Pfenning, we interpret ULL as a session type system for the π-calculus, dubbed πULL. The class of πULL-typable processes therefore contains processes induced by type systems derived from both intuitionistic and classical interpretations of linear logic. Using πULL, we corroborate and formalize Caires, Pfenning, and Toninho's observation as inclusions between classes of typable processes: our technical results are that (i) πULL precisely captures the class of processes typable under the classical interpretation and that (ii) the class of processes typable under the intuitionistic interpretation is strictly included in πULL.

This paper is structured as follows. In Section 2 we introduce ULL, explain the Curry-Howard interpretation as the session type system πULL, and detail the correctness properties for processes derived by typing. Section 3 formally establishes the differences between the classical and intuitionistic interpretations of linear logic as session type systems. Section 4 concludes the paper.

2 United Linear Logic as a Session Type System

In this section, we introduce United Linear Logic (ULL), a logic based on the linear fragment of Girard's Logic of Unity [9]. We present ULL as a session type system for the π-calculus [15, 16], dubbed πULL, following the Curry-Howard correspondences established by Caires and Pfenning [2] and by Wadler [18].

Propositions / Types. Propositions in ULL correspond to session types in πULL; they are defined as follows:

Definition 2.1. ULL *propositions /* πULL *types are generated by the following grammar:*

$$A,B ::= \mathbf{1} \mid \bot \mid A \otimes B \mid A \parr B \mid A \multimap B \mid !A \mid ?A$$

Session types represent sequences of communication actions that should be performed along channels. Table 1 gives the intuitive reading of the interpretation of propositions as session types. Note that there are two types for reception: $\parr$ from classical and $\multimap$ from intuitionistic linear logic.

$\mathbf{1}$ and $\bot$	Close the channel
$A \otimes B$	Send a channel of type A and continue as B
$A \parr B$ and $A \multimap B$	Receive a channel of type A and continue as B
$!A$	Repeatedly provide a service of type A
$?A$	Connect to a service of type A

Table 1: Interpretation of ULL propositions as session types

ULL does not include the additives $A \oplus B$ and $A \,\&\, B$ of linear logic. Although the Logic of Unity does include these connectives, we leave them out from ULL (and πULL), because their interpretation as session types—internal and external choice, respectively—largely coincides in many presentations of logic-based session types. Therefore they are not particularly insightful in our formal comparison. They can be easily accommodated in ULL, with the possibility of choosing between binary choice (as in e.g. [2,4]) and n-ary choice (as in e.g. [3,18]).

Duality. The duality of ULL propositions is given in Definition 2.2. In πULL duality is reflected by the intended reciprocity of protocols between two parties: when a process on one side of a channel sends, the process on the opposite side must receive, and vice versa.

Definition 2.2. *Duality ($A^\bot$) is given by the following set of equations:*

$$\begin{array}{lll} \mathbf{1}^\bot := \bot & (A \otimes B)^\bot := A^\bot \parr B^\bot & (!A)^\bot := ?A^\bot \\ \bot^\bot := \mathbf{1} & (A \parr B)^\bot := A^\bot \otimes B^\bot & (?A)^\bot := !A^\bot \end{array}$$

It is easy to see that duality is an involution: $(A^\bot)^\bot = A$. As usual, we decree that $A \multimap B = A^\bot \parr B$. From this, we can derive the relation between $\multimap$ and $\otimes$ by means of their duals:

$$\begin{array}{l} (A \multimap B)^\bot = (A^\bot \parr B)^\bot = (A^\bot)^\bot \otimes B^\bot = A \otimes B^\bot \\ (A \otimes B)^\bot = A^\bot \parr B^\bot = A \multimap B^\bot \end{array}$$

Processes. πULL is a type system for the π-calculus processes defined as follows:

Definition 2.3. *Process terms are generated by the following grammar:*

$$P, Q := \mathbf{0} \mid [x \leftrightarrow y] \mid (\nu x)P \mid P \mid Q \mid x\langle y\rangle.P \mid x(y).P \mid !x(y).P \mid x\langle\rangle.\mathbf{0} \mid x().P$$

Process constructs for inaction $\mathbf{0}$, channel restriction $(\nu x)P$, and parallel composition $P \mid Q$ have standard readings. The same applies to constructs for output, input, and replicated input prefixes, which are denoted $x\langle y\rangle.P$, $x(y).P$, and $!x(y).P$, respectively. Process $[x \leftrightarrow y]$ denotes a forwarder that "fuses" channels x and y. We consider also constructs $x\langle\rangle.\mathbf{0}$ and $x().P$, which specify the explicit closing of channels: their synchronization represents the explicit de-allocation of linear resources. These constructs result from the non-silent interpretation of $\mathbf{1}$, which, as explained in [3], leads to a Curry-Howard correspondence that is stronger than correspondences with silent interpretations of $\mathbf{1}$ (such as those in [2,4]).

Structural congruence. Processes can be syntactically different, but still exhibit the same behavior. Such processes are *structurally congruent*, in the sense of the following definition:

Definition 2.4. *Structural congruence ($\equiv$) is given by the following set of equations, where $\equiv_\alpha$ denotes equality up to capture-avoiding α-conversion, and $fn(P)$ gives the set of free names in P, i.e. the complement of $bn(P)$: the names in P bound by restriction (νx) and (replicated) input $x(y)$ and $!x(y)$:*

$$P \mid \mathbf{0} \equiv P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$
$$P \mid Q \equiv Q \mid P \qquad [x \leftrightarrow y] \equiv [y \leftrightarrow x]$$
$$(\nu x)\mathbf{0} \equiv \mathbf{0} \qquad P \equiv_\alpha Q \implies P \equiv Q$$
$$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P \qquad x \notin fn(P) \implies P \mid (\nu x)Q \equiv (\nu x)(P \mid Q)$$

Computation. In a Curry-Howard correspondence, computation is related to cut reduction in the logic. Cut reduction removes cuts from an inference tree, which reduces the size of the tree without changing the result of the inference. In the correspondence between linear logic and the π-calculus, cut reduction is related to communication, defined by the following reduction relation:

Definition 2.5. *Reduction of process terms ($\rightarrow$) is given by the following relation:*

$$x\langle y\rangle.P \mid x(z).Q \rightarrow P \mid Q\{y/z\} \qquad Q \rightarrow Q' \implies P \mid Q \rightarrow P \mid Q'$$
$$x\langle y\rangle.P \mid !x(z).Q \rightarrow P \mid Q\{y/z\} \mid !x(z).Q \qquad P \rightarrow Q \implies (\nu y)P \rightarrow (\nu y)Q$$
$$x\langle\rangle.\mathbf{0} \mid x().Q \rightarrow Q \qquad P \equiv P' \wedge P' \rightarrow Q' \wedge Q' \equiv Q \implies P \rightarrow Q$$
$$P \mid [x \leftrightarrow y] \rightarrow P\{y/x\}$$

Typing inference. The inference system of ULL is a sequent calculus with sequents of the form $\Gamma;\Delta \vdash \Lambda$ in which Γ is a collection of propositions that can be indefinitely used, and Δ and Λ collect propositions that must be used linearly (exactly once). With respect to the Logic of Unity, we added a left rule for $\mathbf{1}$ and a right rule for $\bot$, and removed rules that allow propositions to switch sides in a sequent.

πULL's typing inference system annotates sequents with process terms and channel names to form typing judgments of the following form:

$$\Gamma;\Delta \vdash P :: \Lambda$$

In this interpretation, Γ (resp. Δ and Λ), the unrestricted (resp. linear) context of P, consists of assignments of the form $x : A$, where x is a channel and A is a proposition/type. In a correct inference, these contexts together contain exactly the free channel names of the process term P. We write $\cdot$ to denote an empty context. πULL's inference rules are given in Figure 1. Note that some rules are labeled with an '$*$', which we use to distinguish a class of rules to be used in the formal comparison in the next section.

Cut reduction and identity expansion. Caires, Pfenning, and Toninho [3] showed that the validity of session type interpretations of linear logic propositions can be demonstrated by checking that cut reductions in typing inferences do correspond to reductions of processes, as well as by showing that the identity axiom of any type can be expanded to a larger process term with forwarding of a smaller type. Following this approach, πULL can be shown valid for all reductions, using CutR as well as CutL, and expansions, using IdR as well as IdL.

Correctness properties. As is usual for Curry-Howard correspondences for concurrency, the cut elimination property of linear logic means that πULL has the *soundness/subject reduction* (Theorem 2.1) and *progress* (Theorem 2.2) properties.

$$\frac{}{\Gamma; x : A \vdash [x \leftrightarrow y] :: y : A}\ (^*\text{IDR}) \qquad \frac{}{\Gamma; x : A, y : A^\perp \vdash [x \leftrightarrow y] :: \cdot}\ (\text{IDL})$$

$$\frac{\Gamma; \Delta \vdash P :: \Lambda, x : A \qquad \Gamma; \Delta', x : A \vdash Q :: \Lambda'}{\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid Q) :: \Lambda, \Lambda'}\ (^*\text{CUTR})$$

$$\frac{\Gamma; \Delta, x : A \vdash P :: \Lambda \qquad \Gamma; \Delta', x : A^\perp \vdash Q :: \Lambda'}{\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid Q) :: \Lambda, \Lambda'}\ (\text{CUTL})$$

$$\frac{\Gamma; x : A^\perp \vdash P :: \cdot \qquad \Gamma, u : A; \Delta \vdash Q :: \Lambda}{\Gamma; \Delta \vdash (\nu u)(!u(x).P \mid Q) :: \Lambda}\ (\text{CUT}^?)$$

$$\frac{\Gamma; \cdot \vdash P :: x : A \qquad \Gamma, u : A; \Delta \vdash Q :: \Lambda}{\Gamma; \Delta \vdash (\nu u)(!u(x).P \mid Q) :: \Lambda}\ (^*\text{CUT}^!)$$

$$\frac{\Gamma, u : A; \Delta \vdash P :: \Lambda, x : A^\perp}{\Gamma, u : A; \Delta \vdash (\nu x)u\langle x\rangle.P :: \Lambda}\ (\text{COPYR}) \qquad \frac{\Gamma, u : A; \Delta, x : A \vdash P :: \Lambda}{\Gamma, u : A; \Delta \vdash (\nu x)u\langle x\rangle.P :: \Lambda}\ (^*\text{COPYL})$$

$$\frac{}{\Gamma; \cdot \vdash x\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ (^*\mathbf{1}\text{R}) \qquad \frac{\Gamma; \Delta \vdash P :: \Lambda}{\Gamma; \Delta, x : \mathbf{1} \vdash x().P :: \Lambda}\ (^*\mathbf{1}\text{L})$$

$$\frac{\Gamma; \Delta \vdash P :: \Lambda}{\Gamma; \Delta \vdash x().P :: \Lambda, x : \bot}\ (\bot\text{R}) \qquad \frac{}{\Gamma; x : \bot \vdash x\langle\rangle.\mathbf{0} :: \cdot}\ (\bot\text{L})$$

$$\frac{\Gamma; \Delta \vdash P :: \Lambda, y : A \qquad \Gamma; \Delta' \vdash Q :: \Lambda', x : B}{\Gamma; \Delta, \Delta' \vdash (\nu y)x\langle y\rangle.(P \mid Q) :: \Lambda, \Lambda', x : A \otimes B}\ (^*\otimes\text{R})$$

$$\frac{\Gamma; \Delta, y : A, x : B \vdash P :: \Lambda}{\Gamma; \Delta, x : A \otimes B \vdash x(y).P :: \Lambda}\ (^*\otimes\text{L})$$

$$\frac{\Gamma; \Delta \vdash P :: \Lambda, x : B, y : A}{\Gamma; \Delta \vdash x(y).P :: \Lambda, x : A \parr B}\ (\parr\text{R})$$

$$\frac{\Gamma; \Delta, y : A \vdash P :: \Lambda \qquad \Gamma; \Delta', x : B \vdash Q :: \Lambda'}{\Gamma; \Delta, \Delta', x : A \parr B \vdash (\nu y)x\langle y\rangle.(P \mid Q) :: \Lambda, \Lambda'}\ (\parr\text{L})$$

$$\frac{\Gamma; \Delta, y : A \vdash P :: \Lambda, x : B}{\Gamma; \Delta \vdash x(y).P :: \Lambda, x : A \multimap B}\ (^*\multimap\text{R})$$

$$\frac{\Gamma; \Delta \vdash P :: \Lambda, y : A \qquad \Gamma; \Delta', x : B \vdash Q :: \Lambda'}{\Gamma; \Delta, \Delta', x : A \multimap B \vdash (\nu y)x\langle y\rangle.(P \mid Q) :: \Lambda, \Lambda'}\ (^*\multimap\text{L})$$

$$\frac{\Gamma; \cdot \vdash P :: y : A}{\Gamma; \cdot \vdash !x(y).P :: x : !A}\ (^*!\text{R}) \qquad \frac{\Gamma, u : A; \Delta \vdash P :: \Lambda}{\Gamma; \Delta, x : !A \vdash P\{x/u\} :: \Lambda}\ (^*!\text{L})$$

$$\frac{\Gamma, u : A; \Delta \vdash P :: \Lambda}{\Gamma; \Delta \vdash P\{x/u\} :: \Lambda, x : ?A^\perp}\ (?\text{R}) \qquad \frac{\Gamma; y : A \vdash P :: \cdot}{\Gamma; x : ?A \vdash !x(y).P :: \cdot}\ (?\text{L})$$

Figure 1: The ULL inference rules / type system

	Explicit closing	Separate unrestricted context	Identity as forwarding
πCLL [4]	No	Yes	No
CP [18]	Yes	No	Yes
$\pi\mathsf{CLL}^{\mathsf{CP}}$	Yes	Yes	Yes

Table 2: Feature comparison of three session type interpretations of classical linear logic

Theorem 2.1. *If* $\Gamma;\Delta \vdash P :: \Lambda$ *and* $P \to Q$*, then* $\Gamma;\Delta \vdash Q :: \Lambda$.

Proof (sketch). By induction on the structure of the proof of P, using cut reduction. □

Definition 2.6. *For any process P, live*(P) *if and only if there are processes Q and R and channels* $\tilde{n}$ *such that* $P \equiv (\nu\tilde{n})(\pi.Q \mid R)$*, where* $\pi \in \{x\langle y\rangle, x(y), x\langle\rangle, x()\}$.

Theorem 2.2. *If* $\cdot;\cdot \vdash P :: \cdot$ *and live*(P)*, then there exists Q such that* $P \to Q$.

Proof (sketch). The liveness assumption tells us that P is a parallel composition of a process Q that is guarded by some non-persistent communication π and some other process R. The fact that P's proof has an empty context allows us to infer that $\pi.Q$ and R must have been composed using a CUT rule and that R must be guarded by a prefix that is dual to π. Therefore, a reduction can take place. □

3 Comparing Expressivity through United Linear Logic

πULL is a suitable framework for a rigorous comparison of the expressivity of session type systems based on linear logic. In this section, we compare the class of processes typable in πULL to the classes of processes typable in a classical and intuitionistic interpretation of linear logic. For this comparison to be fair, the differences between these classes need to come only from typing, so the process languages need to be the same. This means that our classical and intuitionistic interpretations need to type process terms as given in the previous section, with the same features as those in πULL: explicit closing, a separate unrestricted context, and identity as forwarding.

Our intuitionistic and classical session type interpretations of linear logic are denoted πILL and $\pi\mathsf{CLL}^{\mathsf{CP}}$, respectively. Their respective rules are given in Figure 2 and Figure 3. πILL is based on the work by Caires, Pfenning and Toninho [3].

It is worth noticing that, because of the features we require for our type systems, we could not directly base our classical interpretation on πCLL by Caires, Pfenning and Toninho [4] nor on Wadler's CP [18]. Therefore, we have designed $\pi\mathsf{CLL}^{\mathsf{CP}}$ as a combination of features from πCLL and CP. Table 2 compares these features in πCLL, CP and $\pi\mathsf{CLL}^{\mathsf{CP}}$; the differences are merely superficial:

- Explicit closing of sessions concerns a non-silent interpretation of $\mathbf{1}$ and $\bot$ in the logic that entails a reduction on processes (which, in turn, corresponds to cut reduction). In contrast, implicit closing is due to a silent interpretation and corresponds to (structural) congruences in processes.
- Sequents with a separate unrestricted context are of the form $P \vdash \Gamma;\Delta$, which can also be written as $P \vdash \Delta, \Gamma'$ where Γ' contains only types of the form $!A$.
- The identity axiom can be interpreted as the forwarding process, which enables to account for polymorphism. The forwarding process, however, is not usually present in session π-calculi.

$$\frac{}{\Gamma; x : A \vdash_{\mathsf{ILL}} [x \leftrightarrow y] :: y : A}\ (\textsc{Id}) \qquad \frac{\Gamma, u : A; \Delta, x : A \vdash_{\mathsf{ILL}} P :: z : C}{\Gamma, u : A; \Delta \vdash_{\mathsf{ILL}} (\nu x) u\langle x \rangle.P :: z : C}\ (\textsc{Copy})$$

$$\frac{\Gamma; \Delta \vdash_{\mathsf{ILL}} P :: x : A \qquad \Gamma; \Delta', x : A \vdash_{\mathsf{ILL}} Q :: z : C}{\Gamma; \Delta, \Delta' \vdash_{\mathsf{ILL}} (\nu x)(P \mid Q) :: z : C}\ (\textsc{Cut})$$

$$\frac{\Gamma; \cdot \vdash_{\mathsf{ILL}} P :: x : A \qquad \Gamma, u : A; \Delta \vdash_{\mathsf{ILL}} Q :: z : C}{\Gamma; \Delta \vdash_{\mathsf{ILL}} (\nu u)(!u(x).P \mid Q) :: z : C}\ (\textsc{Cut}^!)$$

$$\frac{\Gamma; \Delta \vdash_{\mathsf{ILL}} P :: z : C}{\Gamma; \Delta, x : \mathbf{1} \vdash_{\mathsf{ILL}} x().P :: z : Z}\ (\mathbf{1}\mathrm{L}) \qquad \frac{}{\Gamma; \cdot \vdash_{\mathsf{ILL}} x\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ (\mathbf{1}\mathrm{R})$$

$$\frac{\Gamma; \Delta, y : A, x : B \vdash_{\mathsf{ILL}} P :: z : C}{\Gamma; \Delta, x : A \otimes B \vdash_{\mathsf{ILL}} x(y).P :: z : C}\ (\otimes\mathrm{L}) \qquad \frac{\Gamma; \Delta \vdash_{\mathsf{ILL}} P :: y : A \qquad \Gamma; \Delta' \vdash_{\mathsf{ILL}} Q :: x : B}{\Gamma; \Delta, \Delta' \vdash_{\mathsf{ILL}} (\nu y) x\langle y \rangle.(P \mid Q) :: x : A \otimes B}\ (\otimes\mathrm{R})$$

$$\frac{\Gamma; \Delta \vdash_{\mathsf{ILL}} P :: y : A \qquad \Gamma; \Delta', x : B \vdash_{\mathsf{ILL}} Q :: z : C}{\Gamma; \Delta, \Delta', x : A \multimap B \vdash_{\mathsf{ILL}} (\nu y) x\langle y \rangle.(P \mid Q) :: z : C}\ (\multimap\mathrm{L}) \qquad \frac{\Gamma; \Delta, y : A \vdash_{\mathsf{ILL}} P :: x : B}{\Gamma; \Delta \vdash_{\mathsf{ILL}} x(y).P :: x : A \multimap B}\ (\multimap\mathrm{R})$$

$$\frac{\Gamma, u : A; \Delta \vdash_{\mathsf{ILL}} P :: z : C}{\Gamma; \Delta, x : !A \vdash_{\mathsf{ILL}} P\{x/u\} :: z : C}\ (!\mathrm{L}) \qquad \frac{\Gamma; \cdot \vdash_{\mathsf{ILL}} P :: y : A}{\Gamma; \cdot \vdash_{\mathsf{ILL}} !x(y).P :: x : !A}\ (!\mathrm{R})$$

Figure 2: The $\pi\mathsf{ILL}$ type system

$$\frac{}{[x \leftrightarrow y] \vdash_{\mathsf{CLL}} \Gamma; x : A, y : A^{\perp}}\ (\textsc{Id}) \qquad \frac{P \vdash_{\mathsf{CLL}} \Gamma, u : A; \Delta, y : A}{(\nu y) u\langle y \rangle.P \vdash_{\mathsf{CLL}} \Gamma, u : A; \Delta}\ (\textsc{Copy})$$

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma; \Delta, x : A \qquad Q \vdash_{\mathsf{CLL}} \Gamma; \Delta', x : A^{\perp}}{(\nu x)(P \mid Q)}\ (\textsc{Cut})$$

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma; x : A^{\perp} \qquad Q \vdash_{\mathsf{CLL}} \Gamma, u : A; \Delta}{(\nu u)(!u(x).P \mid Q) \vdash_{\mathsf{CLL}} \Gamma; \Delta}\ (\textsc{Cut}^?)$$

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma; \Delta}{x().P \vdash_{\mathsf{CLL}} \Gamma; \Delta, x : \bot}\ (\bot) \qquad \frac{}{x\langle\rangle.\mathbf{0} \vdash_{\mathsf{CLL}} \Gamma; x : \mathbf{1}}\ (\mathbf{1})$$

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma; \Delta, y : A \qquad Q \vdash_{\mathsf{CLL}} \Gamma; \Delta', x : B}{(\nu y) x\langle y \rangle.(P \mid Q) \vdash_{\mathsf{CLL}} \Gamma; \Delta, \Delta', x : A \otimes B}\ (\otimes) \qquad \frac{P \vdash_{\mathsf{CLL}} \Gamma; \Delta, y : A, x : B}{x(y).P \vdash_{\mathsf{CLL}} \Gamma; \Delta, x : A \parr B}\ (\parr)$$

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma, u : A; \Delta}{P\{x/u\} \vdash_{\mathsf{CLL}} \Gamma; \Delta, x : ?A}\ (?) \qquad \frac{P \vdash_{\mathsf{CLL}} \Gamma; y : A}{!x(y).P \vdash_{\mathsf{CLL}} \Gamma; x : !A}\ (!)$$

Figure 3: The $\pi\mathsf{CLL}^{\mathsf{CP}}$ type system

Judgments. Before we study the expressivity of these three systems, it is important to take note of the difference in the forms of their typing judgments. For $\pi\mathsf{ULL}$, $\pi\mathsf{CLL}^{\mathsf{CP}}$, and $\pi\mathsf{ILL}$, respectively, they are as follows:

$$\Gamma;\Delta \vdash P :: \Lambda \qquad\qquad P \vdash_{\mathsf{CLL}} \Gamma;\Delta \qquad\qquad \Gamma;\Delta \vdash_{\mathsf{ILL}} x : A$$

$\pi\mathsf{CLL}^{\mathsf{CP}}$ has one-sided sequents, whereas $\pi\mathsf{ILL}$ has two-sided sequents (like $\pi\mathsf{ULL}$), but with exactly one channel/type pair on the right of the turnstile. Compare, for example, $\pi\mathsf{CLL}^{\mathsf{CP}}$'s inference rule for $\otimes$ and the $\otimes$R rules for $\pi\mathsf{ILL}$ and $\pi\mathsf{ULL}$ (see Figure 1):

$$\frac{P \vdash_{\mathsf{CLL}} \Gamma;\Delta,y:A \qquad Q \vdash_{\mathsf{CLL}} \Gamma;\Delta',x:B}{(\nu y)x\langle y\rangle.(P \mid Q) \vdash_{\mathsf{CLL}} \Gamma;\Delta,\Delta',x:A\otimes B}\,(\otimes) \qquad \frac{\Gamma;\Delta \vdash_{\mathsf{ILL}} P :: y:A \qquad \Gamma;\Delta' \vdash_{\mathsf{ILL}} Q :: x:B}{\Gamma;\Delta,\Delta' \vdash_{\mathsf{ILL}} (\nu y)x\langle y\rangle.(P \mid Q) :: x : A\otimes B}\,(\otimes\mathrm{R})$$

Locality. Locality is a well-known principle in concurrency research [13]. The idea is that freshly created channels are *local*. Local channels are *modifiable*, in the sense that they can be used for inputs. Once a channel has been transmitted to another location, it becomes *non-local*, and thus *immutable*: it can only be used for outputs—inputs are no longer allowed. This makes locality particularly relevant for giving formal semantics to distributed programming languages; a prime example is the *join calculus* [6], whose theory relies on (and is deeply influenced by) the locality principle [7].

$\pi\mathsf{ILL}$ guarantees locality for shared channels: a server can be defined using a replicated input, so the channel on which this server would be provided cannot be received earlier in the process. Consider the following example, taken from [4]:

$$(\nu x)(x(y).!y(z).P \mid (\nu q)x\langle q\rangle.Q)$$

Let us attempt to find a typing for P in this process using $\pi\mathsf{ILL}$. We can apply the cut rule to split the parallel composition, of which the left component is $x(y).!y(z).P$. Now, there are two rules we can apply (read bottom-up):

$$\frac{\Gamma;\Delta,y:A,x:B \vdash_{\mathsf{ILL}} !y(z).P :: w:C}{\Gamma;\Delta,x:A\otimes B \vdash_{\mathsf{ILL}} x(y).!y(z).P :: w:C}\,(\otimes\mathrm{L}) \qquad \frac{\Gamma;\Delta,y:A \vdash_{\mathsf{ILL}} !y(z).P :: x:B}{\Gamma;\Delta \vdash_{\mathsf{ILL}} x(y).!y(z).P :: x:A\multimap B}\,(\multimap\mathrm{R})$$

In both cases, the received channel y ends up on the left of the turnstile. There are no rules in $\pi\mathsf{ILL}$ to define a service on a channel on the left and there is no way to move the channel to the right. Hence, we cannot find a typing for P in $\pi\mathsf{ILL}$. In contrast, this process can be typed in $\pi\mathsf{CLL}^{\mathsf{CP}}$ as follows, given $P = P'\{x/u\}$:

$$\dfrac{\dfrac{\dfrac{P' \vdash_{\mathsf{CLL}} \Gamma,u:B;z:A}{!y(z).P' \vdash_{\mathsf{CLL}} \Gamma,u:B;y:!A}\,(!)}{!y(z).P \vdash_{\mathsf{CLL}} \Gamma;y:!A,x:?B}\,(?)}{x(y).!y(z).P \vdash_{\mathsf{CLL}} \Gamma;x:(!A)\parr(?B)}\,(\parr)$$

Symmetry. The rely-guarantee distinction of $\pi\mathsf{ILL}$ is what makes it enforce locality for shared channels. The distinction is visible in $\pi\mathsf{ILL}$'s distinction between left and right in its judgments. Despite the two-sidedness of its sequents, $\pi\mathsf{ULL}$ can also type non-local processes. This is due to the full symmetry in the inference rules: anything that can be done on the left of the turnstile can be done on the right. As we will show formally in the rest of this section, this symmetry corresponds to the single-sidedness of $\pi\mathsf{CLL}^{\mathsf{CP}}$: $\pi\mathsf{ULL}$ and $\pi\mathsf{CLL}^{\mathsf{CP}}$ can type exactly the same processes. We will also show formally that $\pi\mathsf{ULL}$, and thus $\pi\mathsf{CLL}^{\mathsf{CP}}$, can type more processes than $\pi\mathsf{ILL}$, because of the restriction of the right side of the turnstile to exactly one channel/type pair, making $\pi\mathsf{ILL}$ an asymmetrical typing system.

Formal results. Our formal results rely on the sets of processes typable in the three typing systems, given in Definition 3.1.

Definition 3.1. *Let $\mathbb{P}$ denote the set of all processes induced by Definition 2.3. Then*

$$\mathfrak{U} = \{P \in \mathbb{P} \mid \exists \Gamma, \Delta, \Lambda \textit{ such that } \Gamma;\Delta \vdash P :: \Lambda\},$$
$$\mathfrak{C} = \{P \in \mathbb{P} \mid \exists \Gamma, \Delta \textit{ such that } P \vdash_{\mathsf{CLL}} \Gamma;\Delta\},$$
$$\mathfrak{I} = \{P \in \mathbb{P} \mid \exists \Gamma, \Delta, x \in \mathit{fn}(P), A \textit{ such that } \Gamma;\Delta \vdash_{\mathsf{ILL}} P :: x : A\}.$$

Our first observation is that all $\pi\mathsf{ULL}$-typable processes are $\pi\mathsf{CLL}^{\mathsf{CP}}$-typable. Moreover, the reverse is true as well. For the latter fact we need to convert a typing inference with one-sided sequents to a two-sided system, which means that we need the ability to control the side of the turnstile specific propositions end up on. This is taken care of by Lemma 3.1, which is an extension of [4, Prop.5.1, p.19] to include exponential types (! and ?). The main result, Theorem 3.2, is that $\pi\mathsf{CLL}^{\mathsf{CP}}$ and $\pi\mathsf{ULL}$ type exactly the same processes.

Lemma 3.1. *For any typing contexts $\Gamma, \Delta, \Lambda, \Pi$ and process P such that $\Gamma;\Delta \vdash P :: \Lambda, \Pi$, we have $\Gamma;\Delta, \Pi^{\perp} \vdash P :: \Lambda$.*

Proof (sketch). By induction on the structure of the type inference tree. If the last inferred proposition is to be moved to the left, after type inversion, the appropriate left rule can be used where a right rule was used. Other propositions can be moved using the induction hypothesis. □

Theorem 3.2. $\mathfrak{U} = \mathfrak{C}$.

Proof (sketch). There are two things to prove: (i) $\mathfrak{U} \subseteq \mathfrak{C}$, and (ii) $\mathfrak{C} \subseteq \mathfrak{U}$. (i) can be proven by induction on the structure of the typing inferences. The idea is that for every rule of $\pi\mathsf{ULL}$ there is an analogous rule in $\pi\mathsf{CLL}^{\mathsf{CP}}$. As for (ii), for any P such that $P \vdash_{\mathsf{CLL}} \Gamma;\Delta$, it suffices to show that there are $\Delta_L, \Delta_R = \Delta$ such that $\Gamma^{\perp}, (\Delta_L)^{\perp} \vdash P :: \Delta_R$. This can be done by induction on the structure of the typing inference of P. After type inversion, the induction hypothesis can be used, which moves channel/type pairs to either side of the turnstile. This results in many subcases, each of which can be solved with appropriate applications of $\pi\mathsf{ULL}$ rule analogous to the last used $\pi\mathsf{CLL}^{\mathsf{CP}}$ rule, using Lemma 3.1 in some cases. □

The comparison between $\pi\mathsf{ULL}$ and $\pi\mathsf{ILL}$ can be done similarly. However, it is more interesting to examine the set of inference rules. If we restrict all sequents in a $\pi\mathsf{ULL}$ typing inference to have exactly one channel/type on the right of the turnstile, we end up with a subset of usable inference rules: those marked with an '$*$' in Figure 1. Upon further examination, we see that this set of rules coincides exactly with the set of inference rules for $\pi\mathsf{ILL}$. The consequence is that $\pi\mathsf{ULL}$ can type all $\pi\mathsf{ILL}$-typable processes. Finally, any $\pi\mathsf{ULL}$/$\pi\mathsf{CLL}^{\mathsf{CP}}$-typable process violating the locality principle suffices to show the second main result, Theorem 3.3: $\pi\mathsf{ILL}$ is less expressive than $\pi\mathsf{ULL}$. An important corollary of Theorem 3.2 and Theorem 3.3 is that $\pi\mathsf{ILL}$ is less expressive than $\pi\mathsf{CLL}^{\mathsf{CP}}$, confirming the observation by Caires, Pfenning, and Toninho [4].

Definition 3.2 (r-degree)**.** *The size of the right side of a $\pi\mathsf{ULL}$ sequent is the sequent's r-*degree*. Given contexts Γ, Δ, Λ and process P, the ULL sequent $\Gamma;\Delta \vdash P :: \Lambda$ has r-degree $|\Lambda|$.*

Theorem 3.3. $\mathfrak{I} \subset \mathfrak{U}$.

Proof (sketch). Let

$$\mathfrak{U}^* := \{P \in \mathfrak{U} \mid \exists \Gamma, \Delta, A \text{ s.t. } \Gamma; \Delta \vdash P :: x : A \text{ with a proof tree containing only sequents of } r\text{-degree } 1\}.$$

By induction on the structure of the typing inference, it can be shown that all processes in $\mathfrak{U}^*$ have typing inferences using only those rules in Figure 1 marked with an $*$. It follows by contradiction that the last applied rule cannot be without an $*$, and the usage of $*$-marked rules in the rest of the inference follows from the induction hypothesis. The $*$-marked rules with r-degree 1 coincide exactly with the typing inference rules of πILL, and so it follows that $\mathfrak{I} = \mathfrak{U}^*$. Clearly, $\mathfrak{U}^*$ is a subset of $\mathfrak{U}$. Now, given $u : B; \cdot \vdash P :: z : A$, there are several ways to type the process $x(y).!y(z).P$ in πULL, but all of them use sequents of r-degree different from 1, so it is not in $\mathfrak{U}^*$. Hence, $\mathfrak{U}^* \subset \mathfrak{U}$, confirming the thesis. □

4 Conclusion

Using Girard's Logic of Unity [9] as a basis, we have developed United Linear Logic as a means to formally compare the session type systems derived from concurrent interpretations of classical and intuitionistic linear logic. Much as Logic of Unity is a logic that encompasses classical, intuitionistic and linear logic, the session type system interpretation of ULL (dubbed πULL), can type all πCLL$^{\mathsf{CP}}$- and πILL-typable processes. This allows us to compare type systems based on different linear logics.

In πILL, judgments distinguish between several channels whose behavior is being relied on (on the left of the judgment) and a single behavior provided along a designated channel (on the right). To retain this rely-guarantee reading, πULL uses two-sided sequents in its typing inferences. However, πULL does not distinguish between the sides of its sequents; it is fully symmetrical. This symmetry allows it to mimic the single-sidedness of πCLL$^{\mathsf{CP}}$'s sequents: placing a mirror besides πCLL$^{\mathsf{CP}}$'s inference rules reveals the inference rules of πULL. Similarly, πILL can be recovered from πULL by restricting the right side of its sequents to exactly one channel. Not all inference rules remain usable, resulting in an asymmetrical system that coincides exactly with πILL.

Our results formally corroborate the observation by Caires, Pfenning and Toninho: the difference between session type systems based on classical and intuitionistic linear logic is in the enforcement of locality [4]. πCLL$^{\mathsf{CP}}$ is able to type non-local processes, because it does not distinguish between linear channels, received or not, because of its single-sided sequents. Similarly, πULL can type non-local processes because of its symmetry. Restricting πULL into πILL removes exactly those rules needed to violate the locality principle. This reveals that πILL respects locality because of its asymmetry.

Acknowledgements. We are grateful to Joseph Paulus for initial discussions. We would also like to thank the anonymous reviewers for their suggestions, which were helpful to improve the presentation.

References

[1] Andrew Barber & Gordon D. Plotkin (1996): *Dual Intuitionistic Linear Logic*. Technical Report ECS-LFCS-96-347, University of Edinburgh, School of Informatics, Laboratory for Foundations of Computer Science, Edinburgh.

[2] Luís Caires & Frank Pfenning (2010): *Session Types as Intuitionistic Linear Propositions*. In Paul Gastin & François Laroussinie, editors: *CONCUR 2010 - Concurrency Theory*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 222–236, doi:10.1007/978-3-642-15375-4_16.

[3] Luís Caires, Frank Pfenning & Bernardo Toninho (2012): *Towards Concurrent Type Theory*. In: *Proceedings of the 8th ACM SIGPLAN Workshop on Types in Language Design and Implementation*, ACM, pp. 1–12, doi:10.1145/2103786.2103788.

[4] Luís Caires, Frank Pfenning & Bernardo Toninho (2016): *Linear Logic Propositions as Session Types*. *Mathematical Structures in Computer Science* 26(3), pp. 367–423, doi:10.1017/S0960129514000218.

[5] Bor-Yuh Evan Chang, Kaustuv Chaudhuri & Frank Pfenning (2003): *A Judgmental Analysis of Linear Logic*. Technical Report CMU-CS-03-131R, Department of Computer Science, Carnegie Mellon University, doi:10.1184/R1/6587498.v1.

[6] Cédric Fournet, Georges Gonthier, Jean-Jacques Levy, Luc Maranget & Didier Rémy (1996): *A Calculus of Mobile Agents*. In Ugo Montanari & Vladimiro Sassone, editors: *CONCUR '96: Concurrency Theory*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 406–421, doi:10.1007/3-540-61604-7_67.

[7] Cédric Fournet & Cosimo Laneve (2001): *Bisimulations in the Join-Calculus*. *Theoretical Computer Science* 266(1), pp. 569–603, doi:10.1016/S0304-3975(00)00283-8.

[8] Jean-Yves Girard (1987): *Linear Logic*. *Theoretical Computer Science* 50(1), pp. 1–101, doi:10.1016/0304-3975(87)90045-4.

[9] Jean-Yves Girard (1993): *On the Unity of Logic*. *Annals of Pure and Applied Logic* 59(3), pp. 201–217, doi:10.1016/0168-0072(93)90093-S.

[10] Kohei Honda (1993): *Types for Dyadic Interaction*. In Eike Best, editor: *CONCUR'93*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 509–523, doi:10.1007/3-540-57208-2_35.

[11] Kohei Honda, Vasco T. Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In Chris Hankin, editor: *Programming Languages and Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 122–138, doi:10.1007/BFb0053567.

[12] Olivier Laurent (2018): *Around Classical and Intuitionistic Linear Logics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, ACM, Oxford, United Kingdom, pp. 629–638, doi:10.1145/3209108.3209132.

[13] Massimo Merro (2000): *Locality and Polyadicity in Asynchronous Name-Passing Calculi*. In Jerzy Tiuryn, editor: *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 238–251, doi:10.1007/3-540-46432-8_16.

[14] Massimo Merro & Davide Sangiorgi (2004): *On Asynchrony in Name-Passing Calculi*. *Mathematical Structures in Computer Science* 14(5), pp. 715–767, doi:10.1017/S0960129504004323.

[15] Robin Milner, Joachim Parrow & David Walker (1992): *A Calculus of Mobile Processes, I*. *Information and Computation* 100(1), pp. 1–40, doi:10.1016/0890-5401(92)90008-4.

[16] Davide Sangiorgi & David Walker (2003): *The Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press.

[17] Kaku Takeuchi, Kohei Honda & Makoto Kubo (1994): *An Interaction-Based Language and Its Typing System*. In Costas Halatsis, Dimitrios Maritsas, George Philokyprou & Sergios Theodoridis, editors: *PARLE'94 Parallel Architectures and Languages Europe*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 398–413, doi:10.1007/3-540-58184-7_118.

[18] Philip Wadler (2012): *Propositions As Sessions*. In: *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming*, ICFP '12, ACM, Copenhagen, Denmark, pp. 273–286, doi:10.1145/2364527.2364568.